

МАТЕМАТИЧКА ГИМНАЗИЈА

МАТУРСКИ РАД
- из Рачунарства и информатике -

Софтвер отвореног кода и његове примене

Ученик:
Никола Дракулић IVЦ

Ментор:
Мијодраг Ђуришић

Београд, јун 2021.

Садржај

1	Увод	1
2	Софтвер отвореног кода	3
2.1	Историја	3
2.2	Софтвер отвореног кода данас	4
3	Гит	7
3.1	Системи за контролу верзије	7
3.2	Процес рада са гитом	8
3.3	Хостинг пројекта	13
4	Пример контрибуције софтверу отвореног кода	15
4.1	О PowerToys-у	15
4.2	Унутрашње функционисање PowerToys-а	16
4.3	Пријављивање проблема	16
4.4	Идентификација и решавање проблема	18
5	Школско звоно	21
5.1	Хардверског решење	22
5.2	Софтверско решење	22
5.2.1	Страница за контролу звона	23
5.2.2	Извршавање и контрола звоњења	26
5.2.3	Контрола GPIO пинова	28
6	Закључак	29
	Литература	29
7	Додатак	33

1

Увод

Од настанка рачунарских наука траже се начини за ефикасну и једноставну сарадњу између програмера. Лако је закључити да уколико би сваки програмер писао све од почетка, прављење и најједноставнијих програма би захтевало велику количину времена. У овом раду посветићемо се иницијативи отвореног кода (Open Source Initiative - OSI) која је настала 1998. OSI су предводили Jon Hall, Larry Augustin, Eric S. Raymond, Bruce Perens. Размотрићемо историју, алате који се користе као и како се данас тај концепт примењује на све већем броју пројеката где су неки од најзначајнијих Linux, Firefox, MySQL и још многи други.

На примеру ће се објаснити коришћење основних алата, контрибуција пројекту отвореног кода и прављене истог.

2

Софтвер отвореног кода

Организацаија „Иницијатива отвореног кода” је дефинисала софтвер отвореног кода као било који изворни код направљен под једном од лиценци из листе дозвољених лиценци. Неке од најпознатијих лиценци из те листе су следеће:

- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- MIT license
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License version 2.0

Иако лиценце међусобно нису исте све лиценце омогућавају приступ изворном коду неког програма као и његову модификацију, док је разлика на који је начин омогућена даља дистрибуција.

2.1 Историја

У априлу 1991 године, Linus Torvalds, тадашњи студент финског универзитета у Хелсинију, креће да ради на, у том тренутку, малом пројекту који је

требао да буде копија кернела, тадашењег Minix оперативног система под називом Linux. У августу исте године објављује прву верзију програма која је подржавала Bash који је језик за писање скрипти и компајлер за програмски језик C - gcc из GNU пројекта. У његовим описима верзија је забележио да ће Linux подржавати само архитектуру AT-386 како је то био рачунар на коме је радио. Навео је да Linux неће бити ништа професионално као што је то GNU пројекат.

Два месеца касније у верзији 0.02 Linux је могао да покрене већину програма из пројекта GNU. Комплетан код за кернел био је доступан на универзитетском сајту.

База корисника linux-а је махом била састављена од особа које су имале програмерског искуства. Корисници су убрзо почели да користе и да праве своје измене на кернелу које су слали назад Линусу, који је те измене уносио у кернел. Такав начин развијања софтвера био је у противљењу стандардима и навикама већине људи.

GNU пројекат, који је за циљ имао да направи комплетан оперативни систем, издавао је са сваком верзијом цео изворни код, али је методика рада била другачија. Сматрано је да на програму треба да ради мали број људи који ће пажљиво додавати сваку нову могућност. Верзије програма би излазиле на сваких пар месеци.

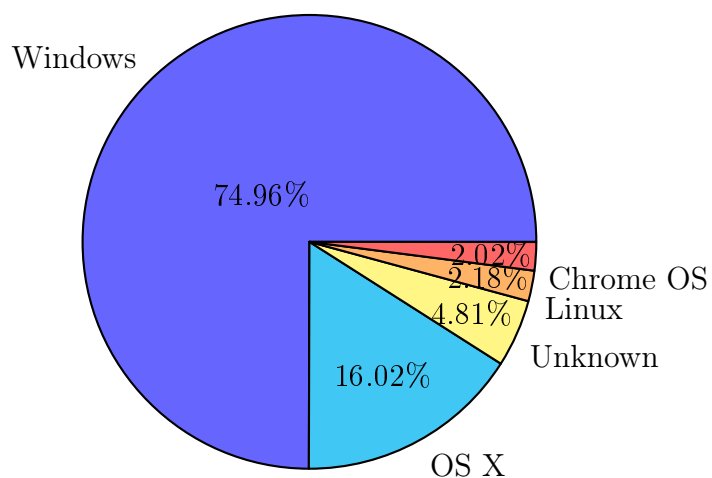
Насупрот томе Торвалдс је прихватио код од великог броја људи и верзије су излазиле више пута недељно. Такав начин рада довео је до много тестера кода који су значајно брже налазили како проблеме тако и решења тих проблема у односу на стил прављења који се до тада користио.

До ових закључака је у свом есеју, под називом „The Cathedral and the Bazaar”, дошао Ерик Раимонд. Рад је изазвао много пажње и довео да компанија Netscape Communications Corporation избаци своју претечу интернет претраживача као бесплатан софтвер. Из њега су се касније развили Firefox - један од највећих претраживача и Thunderbird е-маил клијент за Linux.

2.2 Софтвер отвореног кода данас

Заступљеност апликација отвореног кода је у порасту. Крајњи корисници апликација идаље бирају апликације затвореног кода пре апликација отвореног кода али развијачи софтвера се у све већој мери одлучују за linux и алатке које су отвореног кода.

На персоналним рачунарима linux оперативни систем узима само 2.18% слика (2.1). На серверима је заступљеност linux кернела 75.3% док је на супер рачунарима употпуности заступљен linux.



Слика 2.1: Тржишни удео оперативних система за персоналне рачунаре

Већина мобилних уређаја за оперативни систем има Android(72,2%) који је настао као засебна грана linux кернела коју данас одржава Google. У табели (2.1) можемо да видимо преглед популарних програма у свакодневој употреби као и алтернативе отвореног кода.

Опис програма	Програми затвореног кода	Програми отвореног кода
Пакет програма за канцеларију	Microsoft Office	OpenOffice, LibreOffice
Обрада слика	Adobe Photoshop	GIMP
Векторске графике	Adobe Illustrator	Inkscape
Прављење видео игара	Unity, Unreal Engine	Godot
Инжењерско цртање	AutoCAD	FreeCAD
Интернет претраживачи	Google Chrome	Firefox

Табела 2.1: Преглед алтернатива отвореног кода

3

Гит

3.1 Системи за контролу верзије

У почетку се сва сарадња вршила помоћу размене мејлова и ручног уметања кода других програмера. Временом су се развили системи који помажу да се на систематичан начин развија код и сарађује са више програмера. Такве системе називамо системи за контролу верзија (Version Control System-СКВ). Први програм који предходи модерним СКВ је ИБМ-ов IEBUPDTE направљен 1962 године за архитектуру OS/360. Овај програм је аутоматски надограђивао постојећи код помоћу фајла који је у себи има линије које је потребно измении, слично програму patch на модерним linux системима. 1972 године за исту платформу изашао је програм Source Code Control System (SCCS) који се сматра првим правим системом за контролу верзија. Revision Control System (RCS) замењује SCCS. Предности RCS-а су што је омогућавао контролу пројекта и на мрежи а не само на једном рачунару.

Један од тренутно најпопуларнијих СКВ је Гит. Направљен је од стране Линус Торвалдс-а, 2005 године, као одговор на промене у начину рада BitKeeper-а. Који је до тада коришћен за одржавање Linux кернела. У почетку је то требао да буде само систем ниског ниво који је требао да задовољава следеће ставке:

- Графовска структура
- Не линеарни рад много програмера - могућност да постоје различите гране у верзијама
- Да не користи нови затворени протокол
- Ефикасност на великим пројектима
- Очување интегритета и немогућност да се корумпирају фајлови

- Паковање објеката

Касније су направљене и команде високог нивоа помоћу којих је могуће да се гит користи ефикасно без потребе за додатним програмима

3.2 Процес рада са гитом

Како би смо користили гит у пројекту потребно је да иницијализујемо репозиторијум (за потребе демонстрације име пројекта ће бити project. Линије које почињу са > су линије које bash извршава а линије без > су повратне информације команде)

```
1 > git init
2 Initialized empty Git repository in ~/project/.git/
```

Изворни код 3.1: git init

git init прави у тренутном директоријуму фолдер ./.git у коме се чувају све потребне информације за даљи рад репозиторијума. Додајмо два фајла test.cpp и test.h са следећим садржајем

```
1 void triangle(int);
```

Изворни код 3.2: test.h

```
1 #include<iostream>
2 #include "test.h"
3
4 using namespace std;
5
6 void triangle(int n)
7 {
8     for(int i = 1; i <= n; i++)
9     {
10         for(int j = 1; j <= i; j++)
11             cout << "* ";
12         cout << endl;
13     }
14 }
15
16 int main()
17 {
18     triangle(4);
19 }
```

Изворни код 3.3: test.cpp

Команда `git status` враћа:

- Тренутну грану на којој се налазимо
- Фајлове који су додати у комит
- Промењене фајлове који могу да се додају у комит
- Фајлове који су промењени и које идаље не пратимо

```
1 > git status
2 On branch master
3
4 No commits yet
5
6 Untracked files:
7   (use "git add <file>..." to include in what will be committed)
8     a.out
9     test.cpp
10    test.h
11
12 nothing added to commit but untracked files present (use "git add" to
    track)
```

Изворни код 3.4: `git status`

Приметимо да гит излистава и фајл `a.out` који је компајловани програм. Компајловани фајлови се мењају и не значи нам да чувамо њихове верзије. Гит омогућава да игноришемо фајлове који немају потребе да се чувају прављењем фајла `.gitignore` и записивањем имена фајлове и фолдера које гит треба да игнорише.

```
1 *.out
```

Изворни код 3.5: `.gitignore`

Овим ће гит занемаривати све фајлове који се завршавају са екстензијом `out`.

```
1 > git status
2 On branch master
3
4 No commits yet
5
6 Changes to be committed:
```

```
7 (use "git rm --cached <file>..." to unstage)
8     new file:   .gitignore
9     new file:   test.cpp
10    new file:   test.h
```

Изворни код 3.6: git status

Како би смо сачували измене потребно је да додамо фајлове помоћу команде git add и листа фајлова које желимо да сачувамо. Сада када извршимо команду git status можемо да видимо да имамо три фајла која су спремна за комит. За прављење комита користи се git commit са параметром -m после којег иде опис комита.

```
1 > git commit -m "prvi komit"
2 [master (root-commit) f15df41] prvi komit
3 3 files changed, 21 insertions(+)
4 create mode 100644 .gitignore
5 create mode 100644 test.cpp
6 create mode 100644 test.h
```

Изворни код 3.7: git commit

Из резултата команде можемо да прочитамо следеће. На грани master направили смо први комит са називом први комит. Број f15df41 је скраћени ид комита и помоћу њега можемо да се вратимо на одређени комит. Скраћени ид је настао од првих 7 цифара целог ид-а. Цео ид се генерише као SHA-1 хеш комита. Гит чува све комитове као орјентисани ациклични граф. Подразумевно име главна гране графа је master. Сваки комит поред промена у фајловима чува аутора и време када је направљен комит. Командом git log можемо да видимо све претходне комитове. HEAD је показивач на задњи комит на тренутној грани.

```
1 > git log
2 commit f15df41fb0e4621e26ed3757f10c576c3a9f3482 (HEAD -> master)
3 Author: Drakula44 <ninadrakulic04@gmail.com>
4 Date: Sat May 22 15:34:51 2021 +0200
5
6     prvi komit
```

Изворни код 3.8: git log

Уколико више програмера ради на једном пројекту пракса је да сваки програмер ради на посебној грани како би се избегли конфликти.

Креирање нове гране се извршава командом `git checkout -b` па име гране. Направимо грану `kvadrat` и додајмо могућност да код исписује и квадрате задате величине. Командом `git diff` можемо да видимо промене које смо направили у односу на прошли комит.

```
1 > git diff
2 diff --git a/test.cpp b/test.cpp
3 index 658b2d2..44b083c 100644
4 --- a/test.cpp
5 +++ b/test.cpp
6 @@ -13,6 +13,16 @@ void triangle(int n)
7     }
8 }
9
10 +void square(int n)
11 +{
12 +     for(int i = 1; i <= n; i++)
13 +     {
14 +         for(int j = 1; j <= n; j++)
15 +             cout << "* ";
16 +         cout << endl;
17 +     }
18 +}
19 +
20 int main()
21 {
22     triangle(4);
23 diff --git a/test.h b/test.h
24 index dbc8590..ab917eb 100644
25 --- a/test.h
26 +++ b/test.h
27 @@ -1,2 @@
28 -void triangle(int);
29 \ No newline at end of file
30 +void triangle(int);
31 +void square(int);
32 \ No newline at end of file
```

Изворни код 3.9: `git diff`

Линије кода које испред себе имају `+` су додате док су оне са `-` избрисане. Нека је ово нови комит на грани `kvadrat`. Паралелно је на грани `master` додат комит који омогућава уношење величине троугла из командне линије. Спајање гране `kvadrat` са граном `master` постиже се командом `git merge` па име гране коју спајамо са тренутном. Тако се започиње процес спајања. Гит ће покушати да аутоматски споји све промене где се линије које су промењене не поклапају.

Уколико то није могуће потребно је ручно изабрати промене које се прихватају. Гит ће у фајловима где постоје конфликти убацити све промене па ми можемо да одлучимо које ћемо промене да прихватимо.

```
1 #include <iostream>
2 #include <cstdlib>
3 #include "test.h"
4
5 using namespace std;
6
7 void triangle(int n)
8 {
9     for(int i = 1; i <= n; i++)
10    {
11        for(int j = 1; j <= i; j++)
12            cout << "*";
13        cout << endl;
14    }
15 }
16
17 <<<<<<< HEAD
18 int main(int argc, char *argv[])
19 =====
20 void square(int n)
21 {
22     for(int i = 1; i <= n; i++)
23     {
24         for(int j = 1; j <= n; j++)
25             cout << "*";
26         cout << endl;
27     }
28 }
29
30 int main()
31 >>>>>>> kvadrat
32 {
33     int n = atoi(argv[1]);
34     triangle(n);
35 }
```

Изворни код 3.10: изглед test.cpp фајла који треба ручно да се споји

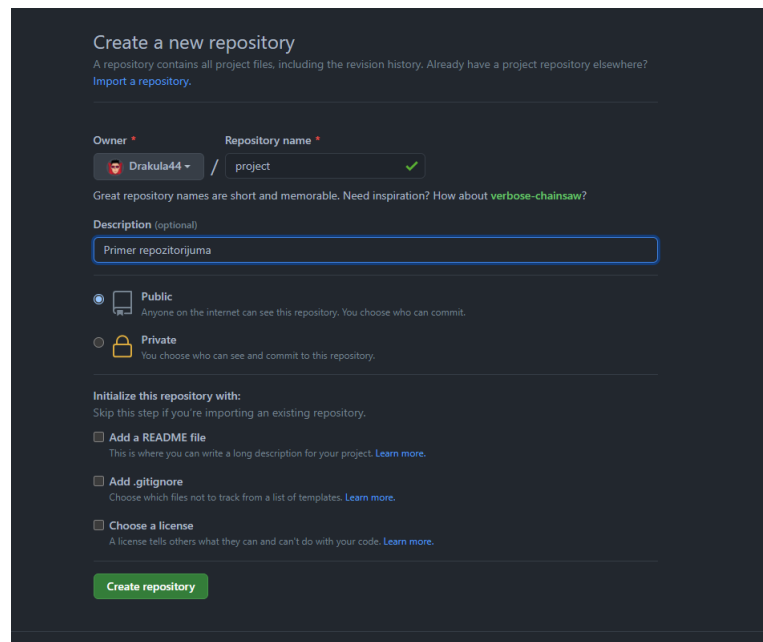
Када извршимо спајање потребно је да направимо комит који ће представљати завршетак спајања. На слици (3.1) је приказан граф комитова који су направљени за потребе ове демонстрације.


```
* 468d1e3 - (12 minutes ago) spajanje - Drakula44 (HEAD -> master)
| \
| * 871c550 - (3 hours ago) dodata funkcija za ispisivanje kvadrata - Drakula44 (kvadrat)
* | 21f5865 - (3 hours ago) unosenje parametra iz komadne linije - Drakula44
| /
* f15df41 - (3 hours ago) prvi komit - Drakula44
```

Слика 3.1: Приказ комитова

3.3 Хостинг пројекта

Како би више људи могло да ради на истом пројекту потребно је да репозиторијум буде на серверу који ће представљати место одакле може да се преузме а касније и поставе надоградње пројекта. На интернету постоји мноштво сајтова који пружају такве услуге. Један од најпопуларнијих је GitHub. Како би смо поставили свој пројекат на GitHub прво је потребно да направимо пројекат на њиховом сајту.



Слика 3.2: Прављење репозиторијума у Github-у

Након направљеног репозиторијума на Github-у потребно је да подесимо удаљени репозиторијум у нашем локалном. Командом

```
1 > git remote add origin https://github.com/Drakula44/project.git
```

Изворни код 3.11: git remote add

додајемо удаљени репозиторијум. Са њим можемо да комуницирамо на три основна начина:

- `git fetch` - помоћу које се проверава које промене постоје на удаљеном репозиторијуму
- `git pull` - преузимамо промене настале на удаљеном репозиторијуму
- `git push` - шаљемо наше надоградње на удаљени репозиторијум

Како на GitHub-у тренутно не постоји ни једна грана потребно је да извршимо команду:

```
1 git push origin —all
```

Изворни код 3.12: `git push`

Origin је назив који смо дали нашем удаљеном репозиторијуму док параметар `—all` говори да се пошаљу промене са свих грана.

4

Пример контрибуције софтверу отвореног кода

У овом делу погледаћемо процес исправљања грешке у коду која је урађена за Microsoft-ову апликацију отвореног кода PowerToys.

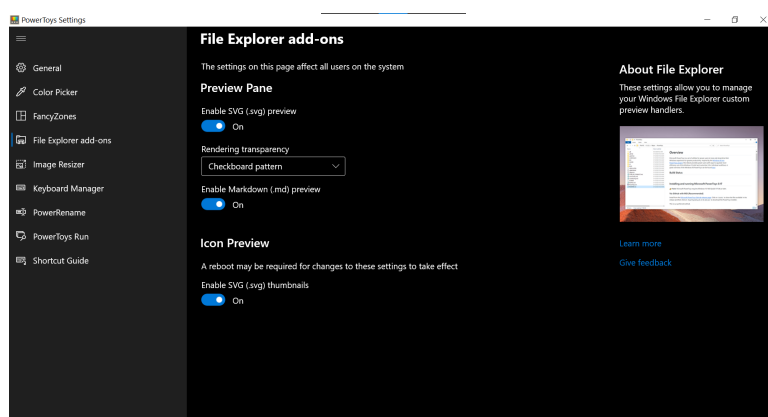
4.1 О PowerToys-у

PowerToys је апликација сачињена од пар под апликација са намером да донесе неке додатне могућности напредним корисницима Windows 10 оперативног система. Листа под апликација:

- Color Picker - алатка за читавање тренутне боје неког пиксела са екрана
- FancyZones - напредна организација отворених прозора и њихово понашање
- File Explorer Add-ons - додатне опције за претраживач датотека File Explorer.
- Image Resizer - алатка у контекст менију за генерисање слика другачијих резолуција
- Keyboard Manager - омогућава замену тастера и пречица.
- PowerRename - алатка за масовно преименовање фајова
- PowerToys Run - мулти функционални претраживачки алат.
- Shortcut Guide - преглед свих пречица које су уграђене у Windows 10 оперативном систему.

4.2 Унутрашње функционисање PowerToys-a

Како би допринели неком коду потребно је да прво разумемо начин функционисања до тадашњег кода. PowerToys има модуларни дизајн. Сваки модул има засебан пројекат и део у корисничком интерфејсу који је писан у C# програмом језику. Засебни делови модула се покрећу као посебни процеси и комуницирају са корисничким интерфејсом помоћу JSON формата. Погледајмо File Explorer Add-ons приказ корисничког интерфејса слика (4.1). Приметимо да се модул састоји из две функције једна додаје додатне типове фајлове које је могуће приказати у панелу за преглед у File Explorer-у, док друга генерише сличице за додатне типове података. Савосталан део модула је написан у C# програмском јер је помоћу њега могуће направити додатне екстензије за File Explorer.



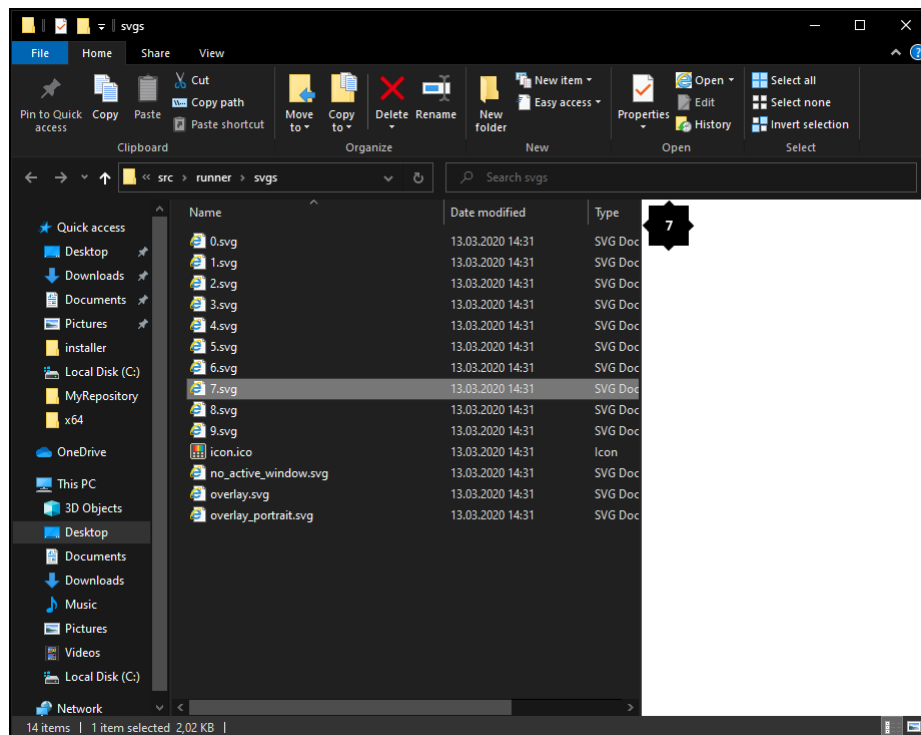
Слика 4.1: Кориснички интерфејс PowerToys-a

Прегледање .svg и .md фајлова је панелу за преглед је реализовано помоћу библиотеке System.Windows.Forms.WebBrowserBase за рендеровање интернет страница.

4.3 Пријављивање проблема

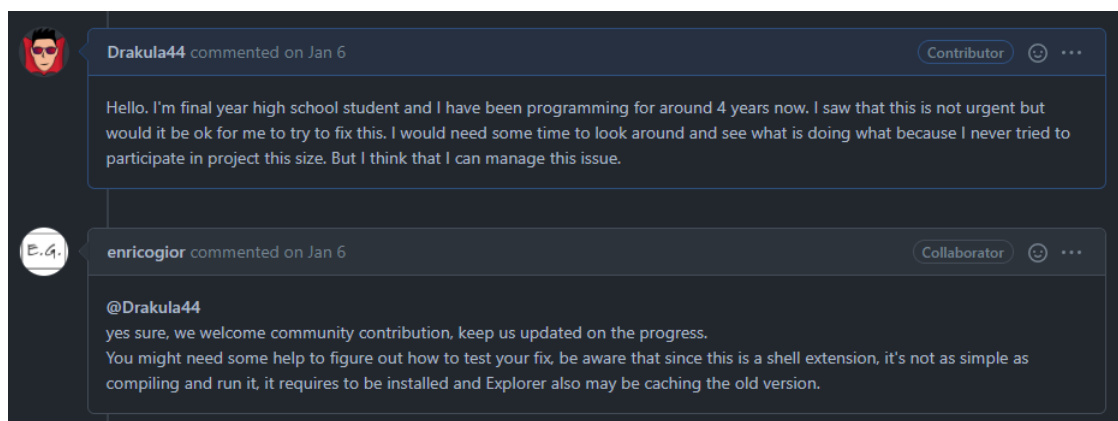
У склопу Github репозиторијума постоји секција Issues у којој пријављени корисници странице могу да пријаве проблеме везане за апликацију.

Корисник PrzemyslawTusinski је приметио да приликом рендеровања .svg фајла приказ се појављивао нецентриран и некалиран слика (4.2).



Слика 4.2: Приказ пријављеног проблема

То није опонашало начин на који је приказивање слика радило у File Explorer-у и обележено је као грешка за који је требала помоћ. Ја сам се јавио да имам жељу да помогнем око ове грешке, слика (4.3).



Слика 4.3: Комуникација са администраторима репозиторијума

4.4 Идентификација и решавање проблема

Како би касније могли да направимо pull request, то јест да затражимо да се наша надоградња програма споји са главном верзијом морамо да направимо fork пројекта на Github-у. Тај fork је потребно преузети локално. У гит-у команда `git clone url` скида пројекат са задате адресе и аутоматски подашева ту адресу као удаљени репозиторијум са називом `origin`.

SVG фајлови су текстуални фајлови који могу да се јаве засебно или у склопу HTML фајлова. Тестирањем се добија да ће интернет претраживачи приказати фајл исто као и у пријављеном проблему али у случају претраживача то је жељени начин. Једно од решења проблема је додавање или уколико већ постоји измена атрибута `style` у коме могу да се дефинишу како ће дати објекат бити рендерован. Скалирање објекта у HTML могуће је постављањем CSS атрибута `max-width` и `max-height` који ће дозволити да висина и ширина објекта буде максимално спецификоване ширине. Центрирање се постиже помоћу атрибута:

```
1 position: absolute;  
2 top: 50%;  
3 left: 50%;  
4 transform: translate(-50%, -50%);
```

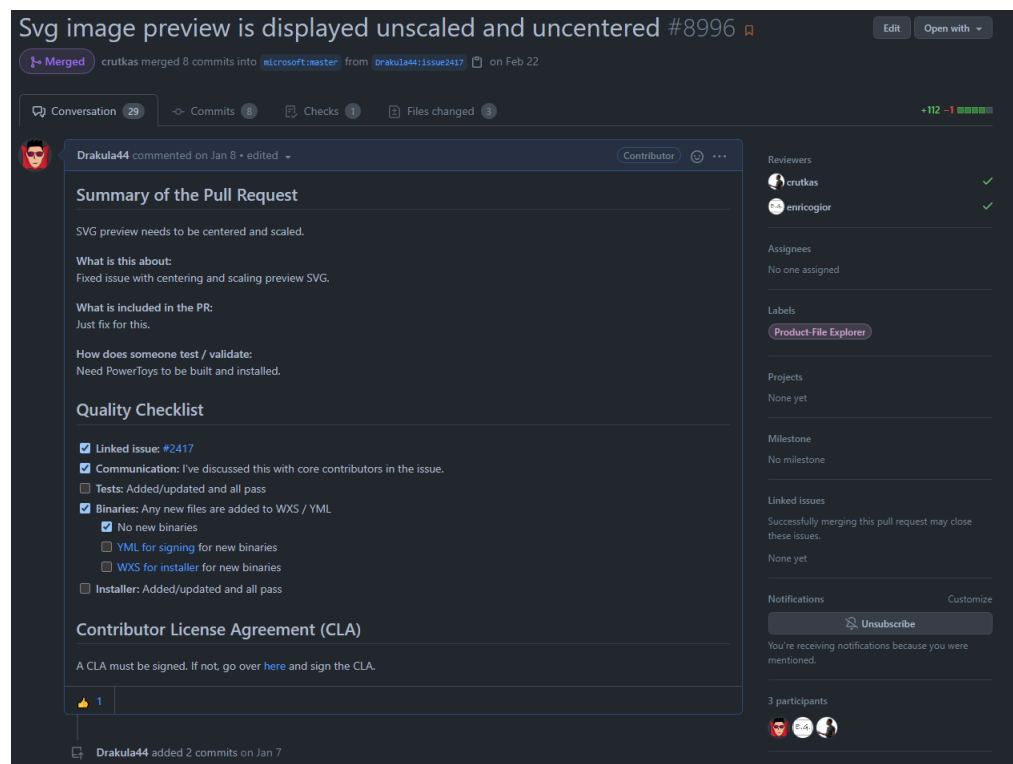
Изворни код 4.1: центрирање елемента

Где ће се прво горња лева ивица поставити на средину екрана и онда ће се атрибутом `transform` објекат транслирати за пола његове ширине лево и пола висине горе. Ово решење је валидно али библиотека која се користи не подржава довољно велику верзију CSS стандарда у коме су се ови атрибути појавили. Библиотека је заснована на Internet Exploreru 8. Атрибути за IE8 који правилно скалирају објекат су следећи:

```
1 _height: expression( this.scrollHeight > {heightR} ? \" {height} \" : \"  
    auto \" );  
2 _width: expression( this.scrollWidth > {widthR} ? \" {width} \" : \" auto \"  
    );
```

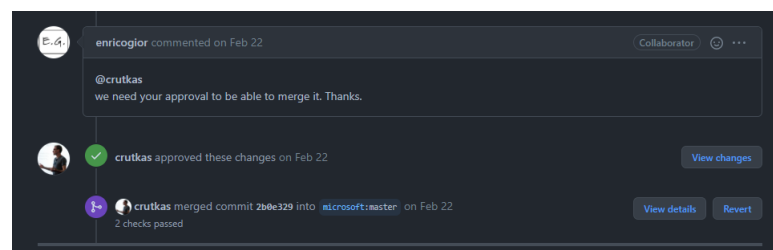
Изворни код 4.2: скалирање елемента

где су `width` и `height` већ налазе међу атрибутима полазног SVG фајла. Након комитовања и слања верзије на лични fork затражује се pull request слика (4.4).



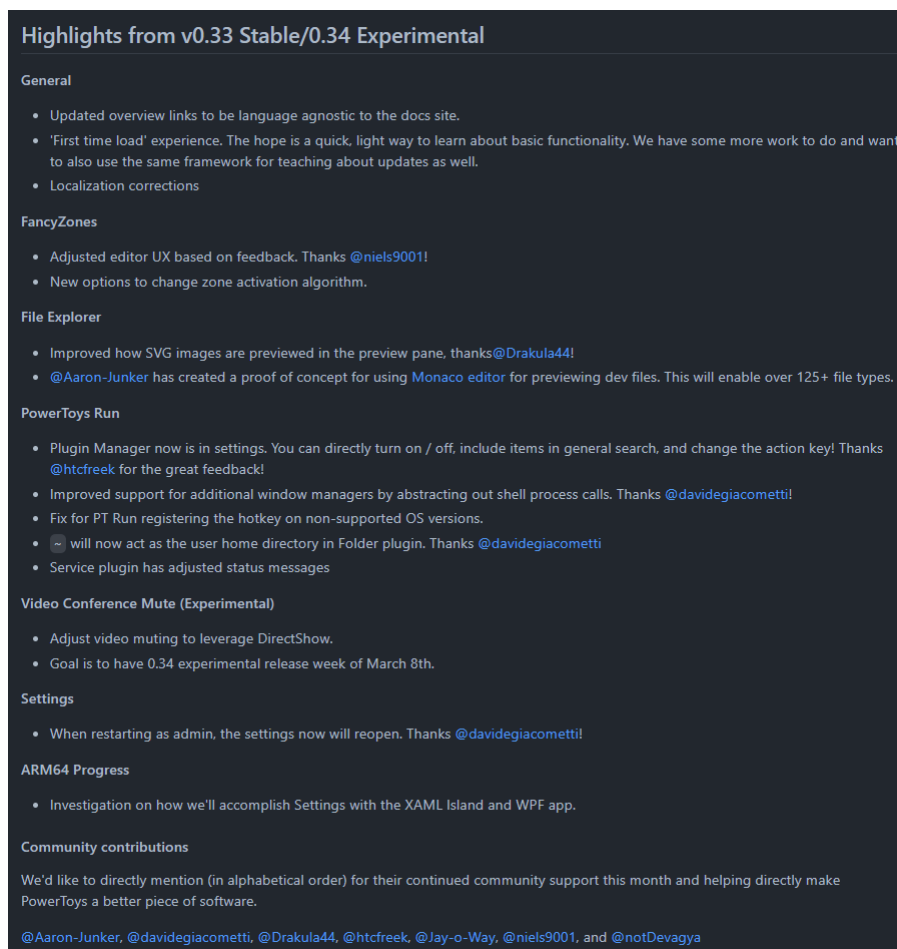
Слика 4.4: Изглед направљеног pull request-а

Након пар интерација над кодом. Уклањања напоменуте грешке у вези са одговарајућом верзијом CSS-а, администратори репозиторијума су спојили моју грану са њиховом главном граном.



Слика 4.5: Успешно спојени pull request

У следећој верзији PoweraToys-а у извештају од аржурирању је био и pull request који сам ја одрадио. Са изашлом новом верзијом се затвара и оригинални проблем.



Слика 4.6: Извештај о промени из верзија када се први пут појављује урађена исправка проблема

5

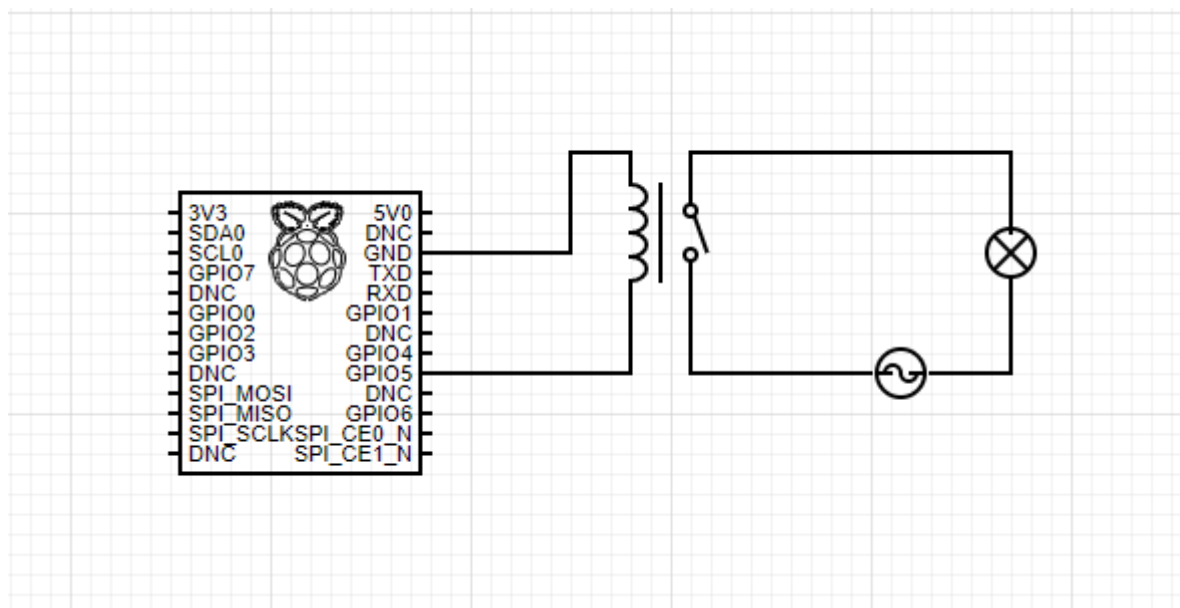
ШКОЛСКО ЗВОНО

Као последњу демонстрацију показаћемо како би требао да изгледа један комплетан пројекат отвореног кода. При креирању репозиторијума на GitHub-у (3.2) могуће је додати лиценцу. За овај пројекат изабрана је MIT License. Она омогућава да било ко може без рестрикција користити код у целини или у његовим деловима. Такође даје обавештење да за овај код не постоји гаранција и да га свако користи на своју одговорност.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



Слика 5.1: Електронска шема школског звона

5.1 Хардверског решење

Пројекат је покренут на Raspberry Pi 3 моделу B. На њему је могуће покренути било коју модерну linux дистрибуцију за ARM архитектуру. Raspberry Pi 3 у себи садржи периферије ниског нивоа помћу којих је могуће софтверски пустити или прекинути доток струје. Напон струје може да буде 0V или 5V. Периферије ниског нивоа се називају GPIO пинови. Како би смо контролисали звоно које је повезано на мрежу од 220V користимо релеј. Релеј је електрична компонента која може да затвори или отвори електрично коло помоћу електро магнета. Довођенем напона од 5v до електро магнета релеј ће затворити коло и покренути звоно. Како би се безбедно приступало апликацији за конфигурацију звона направљена је локална мрежа на коју је повезан Raspberry Pi и особа која треба да подеси звоно слика (5.1).

5.2 Софтверско решење

Пројекат се може поделити на три дела:

- Интернет страница за контролу звона
- Памћење и извршавања задатака - звоњење
- Контрола GPIO пинова

5.2.1 Страница за контролу звона

Контрола звона се врши са странице која је хостована помоћу Flask модула. Како бисмо покренули Flask апликацију потребно је да у командној линији подесимо варијаблу:

```
1 export FLASK_APP=main.py
```

Изворни код 5.1: Подешавање променљивих

где је main.py фајл у коме се налази апликација. Покретање апликације извршава се следећом командом:

```
1 flask run --host=0.0.0.0
```

Изворни код 5.2: flask run

Где параметар host означава да желимо да прихватимо захтеве са свих IP адреса. Апликација ће подразумевно да слуша на порту 5000.

Фолдер у коме се налази Flask апликација мора да поседује одређене фолдере:

- static у њему се чувају сви CSS и Javascript фајлови
- templates у њему су шаблони HTML фајлова писани у Jinja2 формату.

Иницијализација Flask пројекта:

```
1 from flask import Flask
2 from flask import render_template, request, url_for, redirect
3 from flask_assets import Environment
4
5 app = Flask(__name__)
6 assets = Environment(app)
7 assets.register(data.bundles)
```

Изворни код 5.3: иницијализација апликације

Прво направимо главни мени који је на линку / то јест у корену самог сајта. Главни мени садржи два линка. Један је ка страни где можемо да промени тренутни недељни распоред часова. Други помоћу кога палимо или гасимо систем за контролу звона.

```

1 @app.route('/')
2 def index():
3     return render_template('index.html', working=data.working)

```

Изворни код 5.4: Почетна страна

где је index.html шаблон по коме се сајт прави а working промењива да ли тренутно ради звоно.

```

1 <!doctype html>
2 <title>Zvono</title>
3 <ul>
4     <li>
5         <a href="/raspored" style="font-size: 40px;">Promenite raspored
6         </a>
7     </li>
8     {% if working is sameas true %}
9     <a href="/toggle" style="font-size: 40px;">Ugasite zvono</a>
10    {% endif %}
11    {% if working is sameas false %}
12    <a href="/toggle" style="font-size: 40px;">Uplatite zvono</a>
13    {% endif %}
14 </ul>
15 </body>
16 </html>

```

Изворни код 5.5: index.html

Приметимо да Jinja2 формат додаје логику која је по синтакси слича Python програмском језику. Помоћу ње проверавамо да ли систем за управљање звоном ради и исписујемо правилну поруку.

Страна за контролу звона налази се на линку /raspored. Распореду просљеђујемо тренутни недељни распоред као и имена дана у недељи.

```

1 @app.route('/raspored')
2 def weekly_schedule():
3     return render_template('raspored.html', raspored=data.
4                             weekly_schedule, diw=data.days_in_week)

```

```

1 <body>
2 <form action="/submit" method="post" style="display: flex; flex-
  direction: column;">
3   <ul id="nav_section">
4     {% for i in range(7) %}
5       <li id="timeEntry-{{ i }}">
6         <ul>
7           {% if diw[i] %}
8             <h3>{{ diw[i] }}</h3>
9           {% endif %}
10          </ul>
11          {% for j in range(raspored[i]|length) %}
12          <ul id="timeEntry-{{ i }}_{{ j }}">
13            <input onclick="removeRow(this)" type="button"
              value="X" id="timeEntry-{{ i }}_{{ j }}" class=
              "removeButton"><br>
14            Od:<input type="time" id="timeEntryfirst-{{ i }}_{{
              j }}" name="timeEntryfirst-{{ i }}_{{ j }}"
15            value="{{ raspored[i][j][0] }}"><br>
16            Do:<input type="time" id="timeEntrysecond-{{ i }}_{{ j
              }}" name="timeEntrysecond-{{ i }}_{{ j }}"
17            value="{{ raspored[i][j][1] }}">
18          </ul>
19          {% endfor %}
20          <ul id="button-{{ i }}">
21            <input onclick="addRow(this)" type="button" value="Dodajte
              novi cas" id="addNew-{{ i }}">
22          </ul>
23        </li>
24      {% endfor %}
25    </ul>
26    <input type="submit" value="Promenite raspored" class="submit"
      style="font-size: 50px;">
27 </form>
28 </body>

```

Главни део стране распоред представља форма коју генеришемо помоћу постојећег распореда. Промењива `raspored` је листа од 7 листи где је свака листа произвољне дужине. Елементи листе су парови два временена - почетак и завршетак часа. У свакој колони постоји дугме *Додајте нови час* који покреће функцију у JavaScript-у `addRow(el)`

```

1 function addRow(element) {
2   var beforeId = document.getElementById(element.id).parentElement.
    previousElementSibling.id;

```

```

3  var newId;
4  if( beforeId)
5  {
6      newId = beforeId.split("_")[0].concat("_");
7      newId = newId.concat((parseInt(beforeId.split("_")[1])+1).toString
8          ());
9  }
10 else
11 {
12     newId = document.getElementById(element.id).parentElement.
13         parentElement.id.concat("_0");
14 }
15 var firstid = newId.split("-")[0] + "first-" +newId.split("-")[1]
16 var secondid = newId.split("-")[0] + "second-" +newId.split("-")[1]
17 var newEntry = $(`<ul id="${newId}"><input onclick="removeRow(this)"
18     type="button" value="X" id="${newId}" class="removeButton"><br>
19     Od:<input type="time" id="${firstid}" name="${firstid}"><br>Do:<
20     input type="time" id="${secondid}" name="${secondid}"></ul>`);
21 newEntry.insertBefore("#".concat(document.getElementById(element.id).
22     parentElement.id))
23 }

```

Изворни код 5.6: addRow(element)

Функција addRow(element) гледа предходни елемент и генерише ид за нови елемент који ће бити у формату timeEntry-i_j где су i и j позиције елемента у листи. Могуће је избрисати час помоћу функције removeRow(ele) која се покреће притиском на дугме поред.

```

1  function removeRow( element )
2  {
3      parentId = document.getElementById( element.id );
4      console.log( parentId.parentElement )
5      $( "#".concat( parentId.id ) ).remove();
6  }

```

Изворни код 5.7: removeRow(element)

Функција ће избрисати родитељски елемент дугмета то јест један час.

5.2.2 Извршавање и контрола звоњења

За планирање извршавања радњи користимо модул schedule. Како би модул могао да ради не мешајући се са пословима сајта потребно је да га покренемо у другом треду.

```

1 def run_continuously(interval=1):
2     cease_continuous_run = threading.Event()
3
4     class ScheduleThread(threading.Thread):
5         @classmethod
6         def run(cls):
7             while not cease_continuous_run.is_set():
8                 schedule.run_pending()
9                 time.sleep(interval)
10
11     continuous_thread = ScheduleThread()
12     continuous_thread.start()
13     return cease_continuous_run
14
15 stop_run_continuously = run_continuously()

```

Приликом прихваћања форме са /gaspored добијамо python речинк у коме је свако време везано за ид поља из којег је. Помоћу функције dict_to_array конвертујемо тај речинк у листу листи коју остатк програма тражи.

```

1 def dict_to_array(dict):
2     niz = data.weekly_schedule
3     for i in range(7):
4         niz[i] = []
5         j = 0
6         key_first = "timeEntryfirst-" + str(i) + "_" + str(j)
7         key_second = "timeEntrysecond-" + str(i) + "_" + str(j)
8         while key_first in dict and key_second in dict:
9             if dict[key_first] != '' and dict[key_second] != '':
10                 niz[i].append([dict[key_first], dict[key_second]])
11                 j+=1
12                 key_first = "timeEntryfirst-" + str(i) + "_" + str(j)
13                 key_second = "timeEntrysecond-" + str(i) + "_" + str(j)
14     data.weekly_schedule = niz
15     refresh_schedule()

```

На крају функције покрећемо функцију refresh_schedule која ће за свако време и одговарајући дан покренути функцију где је дан из скупа (monday, tuesday, wednesday, thursday, friday, saturday, sunday).

```

1 schedule.every().dan.at(vreme[broj_dana][j][k]).do(ring_bell)

```

Изворни код 5.8: планирање извршавања функције

5.2.3 Контрола GPIO пинова

На почетку програма иницијализујемо пин 17 то јест GPIO пин 6.

```
1 import RPi.GPIO as GPIO
2 GPIO.setmode(GPIO.BCM)
3 GPIO.setup(17, GPIO.OUT)
```

Функциом `ring_bell` постављамо GPIO пин 6 на $5V$ у трајању од 3,5 секунди и након тога га враћамо на $0V$.

```
1 def ring_bell():
2     GPIO.output(17, GPIO.HIGH)
3     time.sleep(3.5)
4     GPIO.output(17, GPIO.LOW)
```


6

Закључак

Циљ рада је био:

- да се покаже развој програмирања отвореног кода кроз време и укаже на његове предности.
- преглед технологије које су потребне да би програмирање отвореног кода било ефикасно.
- Помагање пројекту PowerToys које је помогло корисницима апликације.
- Израда система за школско звоно које у време писања ради у Математичкој гимназији.

Уз предности програмирање отвореног кода, овај приступ има потенцијал да превлада у односу на затворени код али не може скроз да га замени.

PowerToys програм се налази на Github¹-у.

Систем за школско звоно може се наћи на Github²-у као и комплетна документација за код и корисничко упуство. Свака помоћ при унапређењу пројекта је добродошла.

Хтео бих да се захвалим :

- свом ментору Мијодрагу Ђуришићу за помоћ при писању матурског рада
- Enrico Giordani и Clint Rutkas при помоћи при реализацији pull request-a за PowerToys

¹<https://github.com/microsoft/PowerToys>

²https://github.com/Drakula44/skolsko_zvono

Литература

- [1] Raymond, Eric. "The cathedral and the bazaar." Knowledge, Technology & Policy 12.3 (1999): 23-49.
- [2] <https://opensource.org/>
- [3] <https://flask.palletsprojects.com/en/2.0.x/quickstart/>
- [4] <https://jinja.palletsprojects.com/en/3.0.x/>
- [5] <https://pypi.org/project/schedule/>

7

Додатак

Преглед промена које су направљене у pull request-у за пројекат PowerToys

```
1      /// <summary>
2      /// Add proper
3      /// </summary>
4      /// <param name="stringSvgData">Input Svg</param>
5      /// <returns>Returns modified svgData with added style</returns>
6      >
7      public static string AddStyleSVG(string stringSvgData)
8      {
9          XElement svgData = XElement.Parse(stringSvgData);
10
11          var attributes = svgData.Attributes();
12          string width = string.Empty;
13          string height = string.Empty;
14          string widthR = string.Empty;
15          string heightR = string.Empty;
16          string oldStyle = string.Empty;
17
18          // Get width and height of element and remove it afterwards
19          // because it will be added inside style attribute
20          for (int i = 0; i < attributes.Count(); i++)
21          {
22              if (attributes.ElementAt(i).Name == "height")
23              {
24                  height = attributes.ElementAt(i).Value;
25                  attributes.ElementAt(i).Remove();
26                  i--;
27              }
28              else if (attributes.ElementAt(i).Name == "width")
```

```

28         {
29             width = attributes.ElementAt(i).Value;
30             attributes.ElementAt(i).Remove();
31             i--;
32         }
33         else if (attributes.ElementAt(i).Name == "style")
34         {
35             oldStyle = attributes.ElementAt(i).Value;
36             attributes.ElementAt(i).Remove();
37             i--;
38         }
39     }
40
41     svgData.ReplaceAttributes(attributes);
42
43     height = CheckUnit(height);
44     width = CheckUnit(width);
45     heightR = RemoveUnit(height);
46     widthR = RemoveUnit(width);
47
48     string centering = "position: absolute; top: 50%; left:
49         50%; transform: translate(-50%, -50%);";
50
51     // Because WebBrowser class is based on IE version that do
52     // not support max-width and max-height extra CSS is
53     // needed for it to work.
54     string scaling = $"max-width: {width} ; max-height: {height
55         } ";
56     scaling += $"    _height:expression(this.scrollHeight > {
57         heightR} ? \" {height}\" : \"auto\"); _width:expression
58         (this.scrollWidth > {widthR} ? \"{width}\" : \"auto\");
59         ";
60
61     svgData.Add(new XAttribute("style", scaling + centering +
62         oldStyle));
63     return svgData.ToString();
64 }
65
66 /// <summary>
67 /// If there is a CSS unit at the end return the same string,
68 /// else return the string with a px unit at the end
69 /// </summary>
70 /// <param name="length">CSS length</param>
71 /// <returns>Returns modified length</returns>
72 private static string CheckUnit(string length)
73 {
74     string[] cssUnits = { "cm", "mm", "in", "px", "pt", "pc", "
75         em", "ex", "ch", "rem", "vw", "vh", "vmin", "vmax", "%"
76     };

```

```

66         foreach (var unit in cssUnits)
67         {
68             if (length.EndsWith(unit, System.StringComparison.
69                 CurrentCultureIgnoreCase))
70             {
71                 return length;
72             }
73         }
74         return length + "px";
75     }
76
77     /// <summary>
78     /// Remove a CSS unit from the end of the string
79     /// </summary>
80     /// <param name="length">CSS length</param>
81     /// <returns>Returns modified length</returns>
82     private static string RemoveUnit(string length)
83     {
84         string[] cssUnits = { "cm", "mm", "in", "px", "pt", "pc", "
85             em", "ex", "ch", "rem", "vw", "vh", "vmin", "vmax", "%"
86         };
87         foreach (var unit in cssUnits)
88         {
89             if (length.EndsWith(unit, System.StringComparison.
90                 CurrentCultureIgnoreCase))
91             {
92                 length = length.Remove(length.Length - unit.Length)
93                     ;
94                 return length;
95             }
96         }
97         return length;
98     }

```

Извори**код****7.1:**

src/modules/previewpane/SvgPreviewHandler/Utilities/SvgPreviewHandlerHelper.cs

```
1      try
2      {
3          svgData = SvgPreviewHandlerHelper.AddStyleSVG(svgData);
4      }
5      #pragma warning disable CA1031 // Do not catch general exception types
6      catch (Exception ex)
7      #pragma warning restore CA1031 // Do not catch general exception types
8      {
9          _browser.ScrollBarsEnabled = true;
10         PowerToysTelemetry.Log.WriteEvent(new
11             SvgFilePreviewError { Message = ex.Message });
12     }
13     ...
14     _browser.ScrollBarsEnabled = false;
```

Извори код 7.2: src/modules/previewpane/SvgPreviewHandler/SvgPreviewControl.cs